Tutorial

Question 1

In the open addressing schema of Hash table, three probing techniques have been introduced, they are linear probing, quadratic probing, and double hashing. Point out how many different probing sequences for each of the schemes. Compare the advantages and disadvantages of each of the techniques.

What is open addressing? In the open addressing hash table scheme, the *elements are stored in the hash table itself.*

Linear probing

Given a hash function $h'(k) = \{0, 1, \ldots, m-1\}$.

Let $h(k, i) = (h'(k) + i) \bmod m$.

There are $O(m)$ probing sequences because there are m different starting points for the probing and any two probes starting from the same point will have the same sequence.

While linear probing is simple and takes less time, there is the problem of **primary clustering**.

Quadratic probing

Let $h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$.

There are $O(m)$ probing sequences because there are m different starting points for the probing and any two probes starting from the same point will have the same sequence.

Like linear probing, quadratic probing is simple. However, while it avoids the primary clustering problem, there is a problem of **secondary clustering**.

Double hashing

$h(k, i) = (h_1(k) + ih_2(k)) \bmod m$

There are $\Theta(m^2)$ probing sequences because each possible $(h_i(k), h_2(k)$ pair yields a distinct probe sequence. As we vary the key, the initial probe position and offset may vary independently.

Double hashing is an ideal hashing approach. It prevents both primary and secondary clustering problems. However, it is more complicated and requires more running time for hashing.

Question 2

Design a hash function for the open addressing scheme such that it does not suffer from both primary and secondary clustering. In addition, the number of probing sequences derived from it has to be no less than $O(m^3)$ (where m is the number of slots in the hash).

Let $h(k, i) = (h_1(k) + ih_2(k) + i^2h_3(k)) \bmod m$, where h_1, h_2, and h_3 are hash functions that map a key k to one of the m slots available. This scheme avoids both the primary and secondary clustering problems. Also, there are $\Theta(m^3)$ probing sequences as each $(h_1(k), h_2(k), h_3(k))$ 3-tuple yields a distinct probe sequence.

Question 3

Why should the priority queue be introduced? What's its purpose?

The purpose of the priority queue is to maintain a dynamic set on which the following operations can be implemented efficiently: insert, delete, and search (finding the minimum/maximum/priority key).

Question 4

Show how to insert an element to a min-heap such that the heap property holds.

We represent a binary heap as an array of n elements $A[1 \ldots n]$. The parent of any given element i, $P(i) = i/2$. Left child of any given element $L(i) = 2i$, and right child $R(i) = 2i + 1$. To insert an element and maintain the heap property:

1. Insert element into A[n+1].
2. Sift-up from A[n+1]. Call Sift-Up$(A, n + 1)$.

```
def sift_up(A, j):
    i = j
    while i > 1 and A[i] has priority over A[P(i)]:
        swap A[i] and A[P(i)]
        i = P(i)
```

Sift-Up has a runtime of $\Theta(\lg n)$. At the swapping step, we do not need to be concerned about the key of the sibling as $P(i)$ has priority over the sibling. Hence, if i has priority over $P(i)$, then i must have priority over its sibling too.

3. Increment n.